Towards simulation and optimization of cache placement on large virtual Content Distribution Networks

Christos K. Filelis-Papadopoulos[a,b,*], Patricia Takako Endo[c,d], Malika Bendechache[c], Sergej Svorobej[c], Konstantinos M. Giannoutakis[b], George A. Gravvanis[a], Dimitrios Tzovaras[b], James Byrne[c], Theo Lynn[c]

[a]*Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece*
[b]*Information Technologies Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece*
[c]*Irish Institute of Digital Business, Dublin City University, Glasnevin, Dublin 9, Ireland*
[d]*Universidade de Pernambuco, Brazil.*

Abstract

IP video traffic is forecast to be 82% of all IP traffic by 2022. Traditionally, Content Distribution Networks (CDN) were used extensively to meet quality of service levels for IP video services. To handle the dramatic growth in video traffic, CDN operators are migrating their infrastructure to the cloud and fog in order to leverage its greater availability and flexibility. For hyperscale deployments, energy consumption, cache placement, and resource availability can be analyzed using simulation in order to improve resource utilization and performance. Recently, a discrete-time simulator for modelling hierarchical virtual CDNs (vCDNs) was proposed with reduced memory requirements and increased performance using multi-core systems to cater for the scale and complexity of these networks. The first iteration of this discrete-time simulator featured a number of limitations impacting accuracy and applicability: it supports only

[*]Corresponding author

Email addresses: `cpapad@ee.duth.gr`,`cfilpapadop@iti.gr` (Christos K. Filelis-Papadopoulos), `patricia.endo@dcu.ie` (Patricia Takako Endo), `malika.bendechache@dcu.ie` (Malika Bendechache), `sergej.svorobej@dcu.ie` (Sergej Svorobej), `kgiannou@iti.gr` (Konstantinos M. Giannoutakis), `ggravvan@ee.duth.gr` (George A. Gravvanis), `Dimitrios.Tzovaras@iti.gr` (Dimitrios Tzovaras), `james.byrne@dcu.ie` (James Byrne), `theo.lynn@dcu.ie` (Theo Lynn)

tree-based topology structures, the results are computed per level, and requests of the same content differ only in time duration. In this paper, we present an improved simulation framework that (a) supports graph-based network topologies, (b) requests have been reconstituted for differentiation of requirements, and (c) statistics are now computed per site and network metrics per link, improving granularity and parallel performance. Moreover, we also propose a two phase optimization scheme that makes use of simulation outputs to guide the search for optimal cache placements. In order to evaluate our proposal, we simulate a vCDN network based on real traces obtained from the BT vCDN infrastructure, and analyze performance and scalability aspects.

Keywords: vCDN, parallel simulation, cache placement, simulation-based optimization

1. Introduction

Due to advances in wireless and mobile communications, global IP traffic is increasing at unprecedented rates [1]. In 2016, this was predicted to grow threefold by 2021 with the largest portion of this traffic in the form of IP video and more specifically Video-On-Demand (VOD), Internet video, augmented and virtual reality (AR/VR) applications [1]. In order for CSPs to maintain competitive quality of service (QoS) while simultaneously increasing utilization, content distribution networks (CDNs) are used. CDNs distribute content to geographically distributed servers located closer to the users, acting as caches [2, 3]. These caches enhance scalability, reduce latency and limit network congestion, especially in applications such as VOD [4]. According to industry forecasts, CDNs will carry 72% of all internet traffic by 2022 [1].

The increased heterogeneity of devices demanding high resolution video-related content and services (including AR/VR), requires complex networking, increased computing resources, and greater scalability [5]. In order to tackle these demands using legacy infrastructure, resource virtualization is used by CSPs, including virtual CDNs (vCDNs) that leverage cloud computing to offer flexible and reliable services to meet the increasing QoS expectations [6, 7]. Notwithstanding this, use cases, such as video delivery to large audiences, require careful capacity planning and resource management by CSPs to avoid the negative consequences of over- and under-provisioning.

To this end, an improved simulation framework (based on the Discrete Time Simulation (DTS) framework) for virtual Content Distribution Networks proposed in [8]) is presented. This improved framework is combined with a two-phase optimization scheme for finding optimal cache placements. The framework is improved in order to handle more general vCDN architectures described as graphs thereby resulting in more realistic simulations. An improved path formation algorithm is also presented. Moreover, the resource requirements of the requests are decoupled from the available resources on a virtual machine (VM) for a prescribed content type thereby allowing for more general workloads. A finer grained model (per site and per edge) was used for the outputs of the simulation. The updated simulation framework was combined with a novel two-phase cache placement (also called VM placement) optimization scheme for improving functional and non-functional requirements. In the first phase of the proposed approach a number of simulations are conducted with a random number of caches scattered randomly to the available sites. The results of these simulations are weighted and combined with a score function. This score function describes functional and non-functional requirements of the vCDN and results in a ranking for each site. In the second phase, the ranking is utilized in an iterative procedure to discover the optimal number of placements that can service a prescribed number of inputs. The proposed scheme was parallelized for large scale hybrid distributed memory parallel systems using OpenMP and MPI.

In order to demonstrate the applicability of our proposed simulation framework, we simulate a real vCDN architecture using data provided by BT[1], one of the worlds leading communications services companies. Using BT's topol-

[1]https://www.bt.com/

ogy and data for the United Kingdom, we run several experiments in order to assess the effectiveness, scalability and performance of the proposed simulation-optimization technique in conjunction with the updated simulation framework.

In Section 2, the concept of a vCDN is introduced and illustrated through the BT vCDN. This is followed by a brief discussion of simulation-based optimization. In Section 3, the DTS framework for simulating vCDNs is reviewed. Limitations are discussed and proposed solutions for addressing these limitations are presented. In Section 4, the proposed two-phase optimization scheme is presented along with implementation details for large scale hybrid distributed memory systems. In Section 5, the performance and applicability of the updated framework and two-step optimization process is evaluated through a series of simulations at differing scales. Finally, concluding remarks are given and future work is discussed in Section 6.

2. Background

2.1. Virtual Content Delivery Networks

Due to the unprecedented growth in network traffic in terms of volume, velocity, and variety, CSPs are facing challenges to provide competitive quality of service (QoS) levels while achieving greater utilisation of their infrastructures. To improve bandwidth usage, accessibility, and resource utilisation, CDNs are being widely used to distribute content, since they have several cache servers geographically located close to the end users [2, 3].

Tier one network operators, such as BT, host hardware from a variety of different CDN operators. This hardware is deployed at strategic points throughout BT's network which may result in a range of potential problems. These include *(a)* organizing sufficient physical space (in exchange buildings, for instance) to support all of the CDN operators; *(b)* energy to power and cool equipment; and *(c)* discontinuities related to on-boarding and off-boarding CDN operators. Therefore, in order to make better usage of their legacy infrastructure, reduce costs and increase service elasticity, network operators are using virtualization techniques to deploy and offer virtual CDN (vCDN) services. Through virtualization, vCDNs [6, 7] can be dynamically adjusted to meet commercial requirements and associated infrastructure demands.

Herbaut et al. [9] suggest three main drivers of vCDN adoption: *(a)* footprint extension in low density areas where permanent peering may not be economically sustainable; *(b)* quality improvement for niche markets e.g. high-quality content delivery with small delays; and *(c)* providing a dynamic and temporarily increase in bandwidth in a given geographical area (bandwidth bursting). For these reasons, a network operator hosts a vCDN service and thereby replaces multiple customized physical caches with a standard server running multiple virtual applications per CDN operator.

In vCDNs, the main virtual application is the **cache server**. Physical servers of the network operator can host one or more virtual cache servers, from different CDN operators. Each virtual cache server is deployed in the form of a VM or a container.

Optimisation of a vCDN system is a two-stage process. First, the network operator needs to decide where to deploy the physical infrastructure. Second, the CDN operator needs to decide where to install virtual machines or containers that will host the cache server. In this paper, our focus is on cache server placement.

2.1.1. An vCDN hosting example: BT

BT's main activities are the provision of fixed-line services, broadband, mobile and TV products and services, as well as networked IT services. As a service application provider, BT needs to ensure the appropriate QoS for their virtual network functionalities is met.

Currently, 50% of broadband traffic on BTs network originates from a content cache operated by the biggest CDN operators [10]. Today, BT hosts selected CDN Operators' customised cache hardware in two to six nodes in the UK to reduce the amount and cost of Internet peering traffic. If the caches were installed in BTs edge nodes (also known as Tier 1 MSANs (Multi-Service Access

Nodes)), then the cost of delivering content would be reduced by approximately 75% and BT would reduce its network load significantly. However, the CDN Operators (such as Google, Akamai, Limelight, and BBC) are unlikely to want to install their hardware in up to 1,000 locations across the UK for commercially sensitive reasons [10].

Under a vCDN proposition, BT installs the compute infrastructure at its edge nodes and offers a CDN as a Service (CDNaaS). The CDN operators install and manage their own software on the BT CDNaaS, thus maintaining their unique selling points and ownership of their customer base. This is a potential winwin scenario as the network and CDN operators reduce operating costs and consumers get better service.

In this work, we propose a simulation-based optimization solution to estimate optimal cache server placements in a large-scale vCDN infrastructure, using the BT network as our use case.

2.2. Simulation-optimization approach

Traditionally, simulation and optimization techniques have been considered separately, but according to [11], "*tremendous leaps in computational power promoted the appearance of methods that combined both*". Commonly, simulation models are used to understand the system performance under different circumstances and factors, but in some cases system designers or owners, such as network operators, want to explore the impact of different design changes on the overall performance. In this case, the main goal is to perform optimization using simulation models [12].

Simulation-based solutions may rely on a wide set of inputs to feed models and then an optimization approach can be used to optimize this set of inputs (controllable parameter settings). Thus, the simulation and its model detailing, and from the optimization and its ability to find good (or even optimal) solutions, is exploited simultaneously.

Figueira and Almada-Lobo [11] classify the hybrid simulation-optimization approaches in four dimensions: *(i)* simulation purpose, *(ii)* hierarchical structure, *(iii)* search method, and *(iv)* search scheme. The simulation purpose and hierarchical structure are related to the integration between simulation and optimization approaches; while search method and search scheme are related to the search algorithm design. In this paper, we use simulation-optimization as an iterative procedure that uses simulation to evaluate vCDN placement solutions and hence guides the search, validating its decisions.

2.3. Related work

Optimization of cache placement has been extensively studied in the literature, and different approaches have been adopted to formulate cache allocation problems.

For instance, Integer Linear Programming (ILP) has been a popular modeling technique for VM allocation problems [13, 14, 15]. Bari et al. [13] used ILP to formulate VNF orchestration problem (VNF-OP) as an optimization problem which consists in minimizing the cost (deployment cost, energy cost, and cost of forwarding traffic), penalties for Service Level Objective (SLO) violations, and resource fragmentation. However, experimental results from [14] showed that using ILP to find an optimal configuration can result in a very long execution time even for a small number of network functions. The authors thus studied approximations by finding the best fit solution according to a Genetic Algorithm (GA) model of the problem. In particular, they proposed a Genetic Programming-based approach to solve the VM allocation and network management problem by exploring a fixed number of generations. Although their GA may not provide the optimal solution, it can compute configurations orders of magnitude faster than ILP.

The problem of dynamic VM placement was also addressed by the authors in [16, 17] to cope with workloads that change their resource requirements. Based on this work, Yala et al. [18] also adopted GA to address the distribution of the necessary virtual computing resources to a number of virtual service components and the appropriate placement of the latter in the operators NFVI. Here, they captured the conflicting objectives of service availability and deployment cost by proposing a multi-objective optimization formulation for the problem of joint compute resource allocation and VM placement.

Ibn-Khedher et al. [19] proposed two optimization algorithms for both the placement and the migration problem of vCDNs. The major objective from both algorithms is to minimize the total cost of content migration while minimizing the additional costs needed for caching, streaming, and replication number. The first optimization (i.e. $OPAC$: Optimal Placement Algorithm for virtual CDN) [20] formulates an exact algorithm based on a mathematical model for deciding the optimal location to migrate a vCDN or to instantiate (place) a new vCDN on demand to satisfy users quality requirements. Further, to cope with the scalability problem of exact algorithms, the authors adapted a heuristic algorithm (i.e. $HPAC$: Heuristic Placement Algorithm for virtual CDN) to deal with large scale networks.

The approaches discussed above, optimization and simulation are discrete. In the first instance optimization techniques are used to improve resource placement. Then, simulation is used to evaluate the performance of the proposed optimization approach. In contrast, we propose a hybrid simulation-optimization framework that integrates the optimization into the simulation process in order to optimize vCDN placements in a distributed network. The proposed framework is specifically developed to address the scale and complexity of large-scale vCDN deployments. This is accomplished in two ways. Firstly, OpenMP is used to parallelize calculations using shared memory and multiprocessing programming techniques. Secondly, the framework provides support for graph-based network topologies which represent real system network inter-connectivity and hence are more accurate in comparison to more simplistic network depictions. The integration of the simulation framework with two-phase optimization methods creates a novel approach that allows the evaluation of performance, and energy and cost trade-offs during the system design period before full system deployment. To the best of our knowledge, a hybrid simulation-optimization approach has not been used in prior literature for resource placement.

Table 1 summarizes and compares the extant approaches by execution type (parallel or sequential), approach type (optimization only or hybrid simulation-optimization), resource type, scalability, and speedup. This table also highlights the strength of our approach compared to the prior approaches.

Table 1: State of the Art: Comparative Study

	Paper	Execution Type	Approach Type (Goal)	Resource Type	Scalability	Speed Up
ILP approach	[13], [14], [15]	Sequential	Optimization	VM allocation	No	very slow
Meta-heuristic Approach	[14], [16], [17], [18], [19]	Sequential	Optimization	VM placement and network management vCDN placement vCDN migration	No [19]: Yes	Faster than ILP
Exact Approach	[20]	Sequential	Optimization	vCDN placement and Migration	No	Faster than ILP
Our Hybrid Approach	Current paper	Parallel	Simulation-Optimization	vCDN placement	Yes	Fast

3. An improved Discrete Time Simulation framework

Recently, a novel simulation framework for vCDN based on a DTS approach was introduced, [8]. This parallel framework was used to simulate tree-type vCDNs, described by a Directed Acyclic Graph (DAG), where requests were created at the bottom-most level and propagated upwards until a site with a cache to service the request was discovered. An exemplar vCDN infrastructure topology is given in Fig. 1.

Each site is composed of physical host nodes able to accommodate VMs, acting as caches, which serve a specific type of content with prescribed probability. Furthermore, path formation for each request is simplified (due to the tree structure), selecting sites with respect to existence of a cache or the maximum available bandwidth. Each request is characterized only by its duration which is computed using a uniform random number generator between prescribed limits. The resource requirements of each request are derived as a fraction of the available resources on a VM. Thus, requests of the same type have the same resource requirements. The network bandwidth is considered per site and not per connection. The outputs of the simulation framework are collected in a

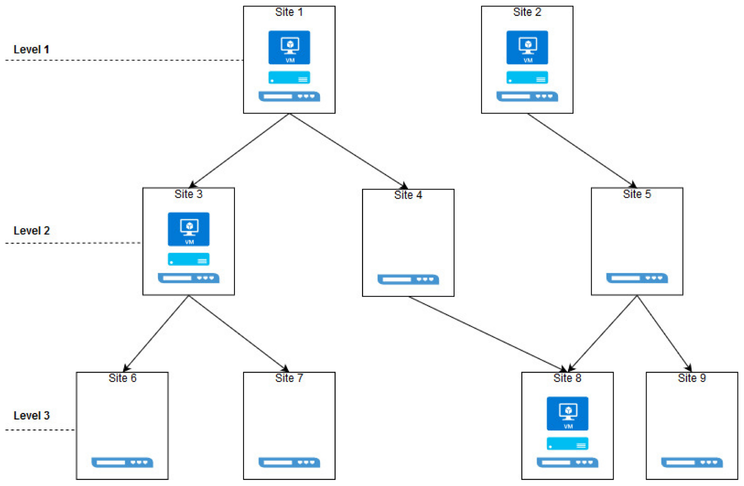


Figure 1: Illustrative vCDN infrastructure topology [8].

statistics engine [8] on a per level basis based on an interval. The framework is parallelized using OpenMP for shared memory parallel systems.

Despite the performance and memory improvements provided by the vCDN simulator proposed in [8], there are a number of limitations. Firstly, the architecture of the vCDN and the connections between sites are stored in an upper triangular sparse adjacency matrix, since connections at the same level are not allowed. However, live vCDNs usually have connections between sites at the same level as well as levels higher than the immediate upper one, especially in cases of isolated sites. This affects the storage of the sites' adjacency matrix. In the case of a tree-type vCDN with lexicographical enumeration of sites from the topmost to the bottom-most level, the adjacency matrix is sparse upper triangular with the diagonal elements indicating each site's corresponding level. By allowing intra-level connections, the resulting sparse adjacency matrix, G, has the following block form:

$$G = \begin{bmatrix} G_{1,1} & G_{1,2} & \dots & \dots & G_{1,\ell} \\ G_{2,1} & G_{2,2} & G_{2,3} & \dots & \vdots \\ \vdots & G_{2,3} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & G_{\ell-1,\ell-1} & G_{\ell-1,\ell} \\ G_{\ell,1} & \dots & \dots & G_{\ell,\ell-1} & G_{\ell,\ell} \end{bmatrix}, \quad (1)$$

where $\ell \in \mathbb{N}$ denotes the number of levels. The matrices $G_{i,i}, 1 \leq i \leq \ell$ retain the connections of each site on level i to other sites at the same level with $diag(G_{i,i}) = i$ the level where each site belongs. The matrices $G_{i,j}, 1 \leq i < j \leq \ell$ retain connections of each site on level i to sites at the lower levels, while matrices $G_{i,j}, 1 \leq j < i \leq \ell$ retain connections of each site on level i to sites at the upper levels. The off-diagonal elements $g_{i,j}$ of matrix G retain the bandwidth (in Gbps) of each connection as opposed to the previous approach where bandwidth was retained as a property of each site. The diagonal elements retain the level of each site. The sparse matrix G describing the structure of the vCDN is generally symmetric. The sparse matrix G is retained in Compressed Sparse Row storage format [21]. For each site i, the sets ν_i and μ_i:

$$\nu_i = \{j : g_{i,j} \neq 0 \cap j \in G_{i,1:i}, j \neq i\} \texttt{ and } \mu_i = \{j : g_{i,j} \neq 0 \cap j \in G_{i,i:\ell}, j \neq i\}, \quad (2)$$

where ν_i denotes the constituent sites at the upper level, including connections to the same level and μ_i the connections to the lower level, including connections to the same level. For each site, data transfer to higher levels is not considered, since it does not substantially affect the vCDN. Thus, each site retains the available bandwidth to constituent sites at the same or higher levels.

The resource requirements of each request are not constant. Instead they are computed based on limits with respect to content type. Each request is described by duration (D), vCPU (P), memory (M), storage (S) and network requirements (N). Thus, each request is described as:

$$r = \{D, P, M, S, N\} \tag{3}$$

with

$$r = \begin{cases} D = & D^{min} & + \left(D^{max} - D^{min}\right) & X_1 \\ P = & P^{min} & + \left(P^{max} - P^{min}\right) & X_2 \\ M = & M^{min} & + \left(M^{max} - M^{min}\right) & X_2 \\ S = & S^{min} & + \left(S^{max} - S^{min}\right) & X_2 \\ N = & N^{min} & + \left(N^{max} - N^{min}\right) & X_2 \end{cases} \tag{4}$$

where X_1 and X_2 are real random numbers in $[0, 1)$ following a uniform random distribution. The requirements $\{P, M, S, N\}$ are computed using the same random number X_2, since they are considered analogous, while duration is independent. Following this approach, available resources are affected differently by each request allowing for more general simulations, instead of the uniform approach followed in [8]. The minimum and maximum values in (4) are defined per content type as different types of content might have different profiles in terms of resource requirements or duration. The duration of a request is measured in *seconds*, while memory and storage in *GB* and network in *Gbps*. The processing requirements are considered as a fraction of a vCPU i.e. if a VM occupies 2 vCPUs then it can host up to 200 requests requiring 0.01 each. Each vCPU corresponds to a CPU.

At each site a request is handled as: *Cached*, *Not Cached*, *Forwarded* or *Rejected* [8]. The first two are valid only in the presence of a cache to the site. When a request is *Cached* it is serviced by the site, occupying P, M, S, N resources, through a path of sites to the user. In case of a *Not Cached* response, a request is forwarded to constituent sites occupying w_1P, w_2M, w_3S, w_4N resources of the site. The variables $w_i \geq 1, 1 \leq i \leq 4$ denote weights and are real numbers greater than 1. A request is forwarded, occupying w_4N resources, when the site has no resources installed in the form of a VM or there are no available resources, except network, in the hosted VMs. A request is *Rejected* when there are no available resources. This approach to handling requests is similar to [8].

In the approach given in [8], the path is formed by at most one site per level. Thus, the maximum number of sites to be traversed until a site hosting a VM that can service a request is discovered is equal to the number of levels ℓ. Moreover, routing of requests could only be performed to the first site at the immediate upper level retaining a VM with sufficient available bandwidth or the site at the immediate upper level with the maximum available bandwidth. However, the inclusion of connections at the same level affects the path formation for servicing a request, since the maximum number of traversed sites is not known *a priori*.

Another major issue is the formation of cycles during path formation. A cycle can be formed when a path of connections exists and sites connect to themselves. In order to avoid formulation of such loops, a dense integer vector, with size equal to the total number of sites, accompanied by an integer list is used for path formation. The elements of the dense vector are zero except those corresponding to the sites already included in the path. The list retains the positions of the nonzero elements of the vector. The dense vector is used to resolve if a site has already been included in $\mathcal{O}(1)$ operations, while the list retains the already visited sites. The items on the list are used to initialize the vector efficiently, avoiding traversal of all elements.

The improved path formation is described algorithmically in Alg. 1. Variations of this algorithm can be used only where maximum bandwidth or maximum bandwidth and cache existence are used for path formation.

In lines $2-3$, initialization of the the set p of the path of sites and the dense vector required to mark visited sites is performed. The algorithm continues by entering the loop to traverse the graph of connections. In line 5 the availability of the current site in terms of resources is checked and in line 6 the site is inserted to path p and marked in v as visited. In case of unavailability the algorithm terminates (lines $7-9$). The connected sites of the current site are traversed and checked based on bandwidth and presence of a cache (lines $12-19$). When

Algorithm 1 Improved Path Formation

```
1:  Let us consider a request $r = \{D, P, M, S, N\}$ entering the system at site $s$
    and $N_s$ the number of sites
2:  $p \leftarrow \emptyset$
3:  $v_i \leftarrow 0, 1 \leq i \leq N_s$
4:  while True do
5:      $t \leftarrow checkAvailability(s, P, M, S, N)$
6:      $p \leftarrow p \cup s, v_s = 1$
7:      if $t =(Rejection)$ then
8:          Return $\emptyset$
9:      end if
10:     if $\nu_s \neq \emptyset$ then
11:         $B \leftarrow 0, s' \leftarrow \emptyset$
12:         for $j \in \nu_s$ with $v_j = 0$ and $s' = \emptyset$ do
13:             if ($r$ is cached in site $j$) and ($availableBandwidth(j) \geq N$) then
14:                 $s' \leftarrow j$
15:             end if
16:             if $B < availableBandwidth(j)$ then
17:                 $B \leftarrow availableBandwidth(j)$,$s \leftarrow j$
18:             end if
19:         end for
20:         if $s' \neq \emptyset$ then
21:             $s \leftarrow s'$
22:         end if
23:     else
24:         Return $p$
25:     end if
26: end while
```

a site is determined it is set as current site and the procedure repeats until the the maximum level is reached (line 10) or a rejection occurs.

The output is stored per site and per connection (edge) for the network as opposed to the aggregated per level approach followed in [8]. Thus, at a prescribed interval, two new files are created corresponding to the state of the sites and the network. This approach enables easier post-processing due to the the multitude of smaller files instead of a very large data output file. Processing a small file, to output data per update interval, can be performed faster than processing one large file leading to improved performance.

In this work, we use this improved vCDN simulation framework as a component for optimizing vCDN cache placements based on simulation outputs.

4. Optimization approaches and strategies via simulation

The improved simulation framework described above has the ability to manage large scale scenarios, as well provide detailed simulation results useful in supporting critical management and operational decisions. In this section, we present our two-phase optimization scheme that makes use of simulation outputs to guide the search for an optimal placement.

4.1. A novel two-phase large scale optimization scheme

Let us consider a vCDN composed of N_s sites arranged in graph with N_s^ℓ number of sites per level, with ℓ denoting the number of levels. The total number of placements for N_s sites is $2^{N_s} \approx 10^{N_s/3.322}$. This number is very large, since in practice a large vCDN is composed of thousands of sites. In order to avoid exhaustive search in the space of possible placements, a two-phase procedure is proposed: (a) to rank the importance of each site, (b) to select the minimum number of placements that satisfies functional and non-functional requirements. The simulation can be described as a process:

$$F(X) \to Y, \; with \; X \in \mathbb{V}^n \; and \; Y \in \mathbb{W}^m, \tag{5}$$

where X is the n inputs and Y the m outputs of the simulation process F. The output vector Y retains metrics extracted from the simulation process. These metrics can be arbitrary in number and represent the quantities that will be used to evaluate the results of the simulation with respect to chosen inputs. The $\mathbb{V}$ and $\mathbb{W}$ are the sets where inputs belong i.e. real numbers. Without loss of generality, let us consider that we want to optimize the system modelled by the process F, for n inputs $X \in \{0,1\}^n$ and m outputs $Y \in [0,1]^m \subset \mathbb{R}^m$. The outputs, obtained with respect to inputs, are assessed by a score function η:

$$\eta(Y) : \mathbb{W}^n \to [0,1] \subset \mathbb{R}, \tag{6}$$

or in the discussed case:

$$\eta(Y) = \sum_{i=1}^{m} w_i Y_i =< w, Y >\in [0,1] \subset \mathbb{R} \tag{7}$$

where w, with $\sum_{i=1}^{m} w_i = 1$, is a vector of weights denoting the importance of each input. In practice η can be any linear or non-linear score function. The score function should be higher (closer to 1) for preferable choices of inputs and lower (closer to 0) in the opposite case.

Let inputs X denote a set of placements of VMs. A value of 1 in a component of X denotes presence of a VM while a value of 0 denotes the absence. Let us consider a number of different placements of N_p formed by a random process with a predefined number of VMs per placement $N_{VM}^j, j \in [1, Np]$:

$$X^j \in \{0,1\}^n, j \in [1, Np] \subset \mathbb{N} \; with \; \sum_i X_i^j > 0 \; and \; X^j \neq X^k, j, k \in [1, Np], \tag{8}$$

and corresponding outputs and scores:

$$Y^j = F(X^j) \in \mathbb{R}_+^m \; and \; \eta(Y^j) = \eta_j, \; j \in [1, Np] \subset \mathbb{N}. \tag{9}$$

By combining the outputs weighted by the score we have:

$$X^o = \frac{1}{N_p} \sum_{j=1}^{N_p} \eta(Y^j) \frac{X^j}{\|X^j\|_\infty} = \frac{1}{N_p} \sum_{j=1}^{N_p} \frac{1}{\|X^j\|_\infty} \eta_j X^j \in [0,1]^n \subset \mathbb{R}^n. \tag{10}$$

where $\|x\|_\infty = \max\limits_i |x_i|$ denotes the infinity (or maximum) norm of a vector x.

The outputs that yield improved results will influence the output more, due to the score function. Moreover, VMs participating in more than one random placements, with improved results, will have a greater value in the vector X^o. In the limit case, when $N_{VM} = 0$ the vector $X^o = \vec{0}$.

The vector X^o expresses the importance of each placement of a VM to a site, with respect to the number of inputs. The components $i \in \{X_i^o \geq \alpha \ : 1 \leq i \leq N_s, \alpha \in [0,1] \subset \mathbb{R}\}$ denote the $|i|$ the most important placements. This can be interpreted also as the probability of a placement being important, with values close to 1 denoting most probable placement with probability equal to X_i^o, $1 \leq i \leq N_s$. The components of the vector associated with placements on the topmost level are explicitly set to 1, since sites in this level always retain a VM (cache).

This approach can be used for online assessment of placements, since a change in the distribution of input requests to sites will affect the effectiveness of placements, designating new ones as improved. The new placements might designate different sites as more appropriate, weighting the remaining sites with a reduced score and due to continuous division reducing their significance within a few timesteps. The update formula for vector X^o with a new placement X^{N_p+1}, with $Y^{N_p+1} = F(X^{N_p+1})$ and $N_{VM}^{N_p+1}$ number of VMs, is the following:

$$\begin{aligned} (X^o)^{N_p+1} &= \frac{N_p\,(X^o)^{N_p} + \frac{1}{\|X^{N_p+1}\|_\infty}\eta\left(Y^{N_p+1}\right)X^{N_p+1}}{N_p+1} \\ &= \frac{N_p\,(X^o)^{N_p} + \frac{1}{\|X^{N_p+1}\|_\infty}\eta_{N_p+1}X^{N_p+1}}{N_p+1} \end{aligned} \tag{11}$$

where $(X^o)^{N_p}$ denotes the vector X^o after N_p possible placements.

In order to compute the minimum number of placements required to efficiently service a prescribed workload the distribution of importance of placements given by the vector X^o can be used. Let us consider a procedure $\Sigma(X^o, I) : (\mathbb{R}^n, \mathbb{N}^n) \to (\mathbb{R}^n, \mathbb{N}^n)$, which sorts the components of a vector in descending order and returns the indices of the sorted elements with respect to initial ordering. The vector $I = [1\ 2\ \ldots\ N_s]$ retains the initial ordering. The vectors $\left(\hat{X}^o, \Xi\right) = \Sigma(X^o, I)$ retain possible placements in order of importance and the corresponding set of indices Ξ. The set Ξ is used to form a new set of placements $V_j, 1 \leq j \leq N_p^* \leq |\Xi|$ such that:

$$V_j(\xi) = 1,\ \xi = \Xi_{1:j},\ 1 \leq j \leq N_p^*, \tag{12}$$

where N_p^* is the maximum allowed number of VMs (caches) that can be deployed. The simulations are performed in sequence with the new placements $V_j,\ Z_j = F(V_j)$ until a minimum number of placements is found that satisfies prescribed functional and nonfunctional requirements i.e. no rejections of requests and minimum energy consumption. The placement that satisfies these requirements is considered as the output of this two-phase optimization scheme. Where a minimum number of placements is computed, the remaining simulations are not executed. Moreover, where sites always retain VMs, such as those on the topmost levels, the corresponding sites are placed first with $X_i^o = \hat{X}_i^o = 1$ and $I_i = \Xi_i$ with $1 \leq i \leq \kappa$, where κ denotes the first sites always retaining a VM.

The proposed two-phase optimization procedure can be used for any other simulation framework F, since it does not depend on parameters related to the DTS approach.

4.2. Implementation details

The proposed two-phase optimization scheme requires an increased number of simulations to be performed. In order to accelerate ranking the placements and finding the most appropriate one, the scheme is parallelized for distributed memory systems with multicore nodes to exploit the parallelism of the simulation framework. Initially, the random placements are created at each distributed node. Each node forms N_p random placement vectors with respect to a maximum number of VMs N_{VM}. Where the maximum number of VMs is not prescribed, it is set equal to the number of available sites N_s. The number of VMs per placement is $N_{VM}^{j}, 1 \leq j \leq N_p$. The number of VMs per placement and the corresponding positions are computed randomly using a non-deterministic random number generator based on stochastic processes [22]. This type of random generator is chosen to avoid the creation of the same placement in different nodes leading to biased results. Multiple instances of the same placement will influence the final ranking vector towards this placement, especially if this placement has a high score, since the final ranking vector is computed by averaging local score vectors. The use of a non-deterministic random generator limits this phenomenon since random numbers are computed based on an entropy pool collecting environmental noise from device drivers and other sources [22]. The formed VM placements are saved to files stored in the shared storage. Each node performs all simulations and stores results in the shared storage:

$$Y_{(r)}^{j} = F\left(X_{(r)}^{j}\right), \ 1 \leq j \leq N_p, \ 1 \leq r \leq N_n, \tag{13}$$

where r is the rank of each distributed node and N_n denotes the number of nodes. The total number of simulations is equal to $N_p \, N_n$. The simulations are performed on the multicore nodes in parallel using OpenMP, since the framework is designed to be parallel. For sufficiently large numbers of simulations per node N_p the workload is expected to be balanced since the distribution of random numbers is selected to be uniform. Each node computes the score based on the outputs of each simulation. The score function is as follows:

$$\eta\left(Y\right) = \sum_{i=1}^{3} \beta_i f_i, \tag{14}$$

where f_i are functions describing several performance metrics and β_i are the weights with $\sum_{i=1}^{3} \beta_i = 1$. For the j-th simulation, we have:

- $f_1(u) = \frac{1}{N_{VM}^{j}} \sum_{i=1}^{N_{VM}^{j}} u_i$: The average vCPU utilization of the VMs of the j-th placement. This function is used to assess the over-provisioning of a placement.

- $f_2(N_{VM}^{j}) = \frac{1}{1+e^{-5+10N_{VM}^{j}/N_s}}$: The number of VMs of the j-th placement.

This function is used to penalize placements with very large number of VMs. The function is based on the sigmoid function, penalizing more aggressively placements that include VMs in more than 50% of the available sites.

- $f_3(A^j, R^j) = \frac{R^j}{A^j + R^j}$: The number of accepted requests A^j and the number of rejected requests R^j for the j-th placement. This function is used to penalize the score in case of rejected requests. It also increases the value of the score when placements with large number of VMs are required, acting as a counterbalance to function f_2.

The weights are chosen to be $\vec{\beta} = [0.2\ 0.1\ 0.7]$. The weights are chosen in such an order to penalize the presence of rejected requests more thus favoring placements that lead to less rejected requests more. The score for each placement $\eta\left(X_{(r)}^j\right) = \eta_j^{(r)}$ is computed and multiplied by the respective placement vector. The result for each node r is computed as follows:

$$X_{(r)}^o = \frac{1}{N_p} \sum_{j=1}^{N_p} \eta\left(X_{(r)}^j\right) X_{(r)}^j = \frac{1}{N_p} \sum_{j=1}^{N_p} \eta_j^{(r)} X_{(r)}^j. \tag{15}$$

Each node should communicate the result vector $X_{(r)}^o$ to the head node, where all $X_{(r)}^o$ are accumulated and the result is averaged:

$$X^o = \frac{1}{N_n} \sum_{r=1}^{N_n} X_{(r)}^o. \tag{16}$$

The vector X^o is sorted, resulting in the sorted ranking vector $\hat{X}^o$ and the sorted indices vector Ξ. The vector of indices is broadcast to all nodes. In the case of the topmost level, where the sites always retain a VM, the elements corresponding to these sites are not considered during sorting, thus $\hat{X}_i^o = X_i^o = 1$ and $\Xi_i = I_i$ with $1 \leq i \leq \kappa$ and κ denoting the number of sites at the top-most level.

The second phase requires the execution of multiple simulations to determine the optimal number of placements to service the prescribed amount of requests.

Since, the performance of the simulation is affected by the number of placements a different approach is used to ensure load balancing. Each node is assigned to execute simulations with number of placements equal to multiples of the total number of nodes starting from its rank. Thus, each node computes simulations with number of VMs equal to:

$$N_{VM}^{j} = \kappa + r + (j-1)N_n, j \in \mathbb{N}, 1 \leq j \leq 1 + \left\lfloor \frac{N_s - r - \kappa}{N_n} \right\rfloor, \tag{17}$$

or in case of a prescribed maximum number of VMs:

$$N_{VM}^{j} = \kappa + r + (j-1)N_n, j \in \mathbb{N}, 1 \leq j \leq 1 + \left\lfloor \frac{N_{VM} - r - \kappa}{N_n} \right\rfloor. \tag{18}$$

The placements of the VMs are chosen from the rankings obtained during the first phase. The iterative process may terminate earlier if a new placement yields worse results than a previous simulation with respect to a score function. This score function is based only on the number of rejected tasks. Thus, the new score function is $\theta(Y) = \theta(A, R) = \frac{A}{A+R}$, where A denotes the number of accepted requests and R denotes the number of rejected requests. However, this is a strict termination criterion, that may lead to premature termination of the optimization process especially in cases where successive simulations yield almost similar scores due to slight differences in the distribution of inputs. These differences arise from the use of the non-deterministic uniform random generator for constructing the sequence of input requests along with their requirements. Thus, continuation of experiments is allowed when the produced score is greater than $\delta\theta(Y)$. This is a more relaxed supplementary criterion. The value of the parameter $\delta \in [0, 1) \subset \mathbb{R}$, results in stricter criterion when closer to the unit and a more relaxed one when closer to zero. However, relaxing the termination criterion might result to stagnation of the iterative process leading to an increased amount of iterations that do not improve the final placement. Thus an upper bound $M_\delta \in \left[1, 1 + \left\lfloor \frac{N_{VM} - r - \kappa}{N_n} \right\rfloor \right] \subset \mathbb{N}$ on the allowed stagnation steps has to be enforced.

Each executed placement is written to a file on the shared storage. When the execution of the simulations terminates, the score of the best placement per node is sent to the head node. The head node selects the maximum score and obtains the corresponding placement from the node that achieved it. This optimal placement is stored into a file on the shared storage. The proposed two-phase optimization scheme is described in Alg. 2. In Alg. 2, parameter κ denotes the number of sites that always retain a placement, N_p the number of placements per node, N_n the number of nodes, r the rank of a distributed node, N_s the number of sites, N_{VM} the maximum number of allowed placements and M_δ is the upper bound on the stagnation steps allowed. The vector X^{opt} denotes the optimal placement with respect to the chosen score function.

In lines 1 of Alg. 2, the vector retaining the sum of scores and placements X_r^o is initialized. In lines $2-4$ all simulations are executed on all available nodes after creating random input placements (line 3) and results of placements multiplied by scores are locally aggregated. Following (line 6), the aggregated vector is normalized by the number of simulations N_p and send to head node. The head node receives all vectors and aggregates them (line 7). Then, the aggregated vector X_1^o is sorted to produce the ranking Ξ (lines $8-11$) and is broadcast back to the nodes (line 12) in order to start the second phase. The parameters corresponding to local optimal score (θ_r) based on rejected tasks, local optimal number of placements π_r and local stagnation step counter (c_δ) are initialized (line 13). The local nodes start to execute simulations based on the given ranking (lines $15-16$) with increasing number of placements, retaining the maximum score attained and its corresponding number of placements (lines $17-18$), until an optimal value has been discovered or stagnation has occurred (lines $19-25$). The number of placements is always greater or equal κ, since the nodes at the topmost level are always first in ranking and are considered to always retain a cache (line 14). The local optimal numbers are sent to the head node where the global maximum score is computed and the corresponding number of placements is recovered (line 27). With this information the optimal placement is formed by the head node (lines $28-30$).

The communication required by the proposed scheme are limited to one reduction operation, two broadcast operations, and a send-receive operation. The reduction operation is performed after the end of the first phase in order to collect all local score vectors and form the global ranking vector. This operation is followed by a broadcast operation to transfer the global score vector to all compute nodes after sorting has been performed on the head node. Finally, after the end of the second phase, the maximum scores are gathered from the compute node and a flag is broadcast in order to inform the node retaining the maximum score to send the number of caches required for the optimal placement. This number is used to derive the optimal placement from the ranking vector.

These operations do not affect substantially the performance of the proposed scheme. All distributed computations are performed using the Message Passing Interface (MPI). In Fig. 2 a schematic representation of the two-phase scheme is given.

5. Simulation results

5.1. Characteristics of the vCDN and parameters of the simulations

The vCDN topology provided by BT and used for assessing the proposed simulation and optimization approach is composed of four levels (top-most to bottom-most): core sites (with two levels, inner-core and outer-core), metro sites, and Multi-Service Access Nodes (MSAN) sites (composed of tiers 1, 2, and 4), as shown in Figure 3.

The graph describing the network of sites has 3056 edges corresponding to connections between the 1132 vertices of corresponding sites. The basic characteristics of the vCDN are given in Tab. 2. The total connections correspond to connections of a site with sites at the same or the immediate lower level.

The number of connections per site increases closer to the top of the hierarchy compared to lower levels in the hierarchy. Similarly, the bandwidth per connection and per site increases higher in the hierarchy. This is expected since sites in higher levels service an increased number of requests forwarded from

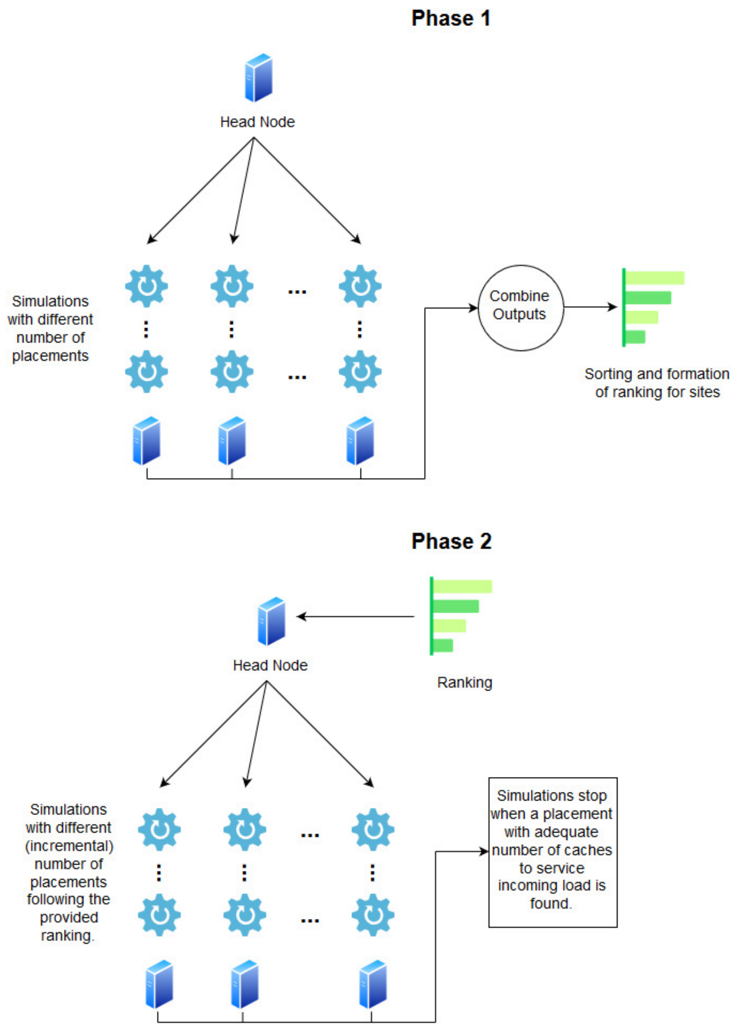

Phase 1
Head Node
Simulations with different number of placements
Combine Outputs
Sorting and formation of ranking for sites
Phase 2
Head Node
Ranking
Simulations with different (incremental) number of placements following the provided ranking.
Simulations stop when a placement with adequate number of caches to service incoming load is found.

Figure 2: High level schematic representation of the proposed two-phase optimization scheme.

sites at the immediate lower level. The topmost level has the most bandwidth per connection as well as the most connections, since sites in this level retain all content and can service any request that is forwarded. The number of connections at level 4 corresponds to intra-level connections of sites. Level 3 has no intra-level connections, thus all its connections are to sites in the immediate lower level (level 4). The number of connections of level 3 to level 4 is substantially less compared to the number of sites at that level. Thus, a lot of sites in

Algorithm 2 Two-phase optimization scheme

```
1: Set $X_r^o \leftarrow \vec{0}$
2: for $j \in [1, N_p]$ do
3:     Form $X^j$
4:     Compute $Y^j \leftarrow F(X^j)$, $\eta_j \leftarrow \eta(Y^j)$, $X_r^o \leftarrow X_r^o + \eta_j X^j$
5: end for
6: Compute $X_r^o \leftarrow \frac{1}{N_p} X_r^o$
7: Reduce $X_1^o \leftarrow \sum_{r=1}^{N_n} X_r^o$ to node $1$
8: if r=1 then
9:     Initialize $I = [1, 2, \cdots, N_s]$
10:    Compute $X_1^o \leftarrow \frac{1}{N_n} X_1^o$, $\left(\hat{X}^o, \Xi\right) = \Sigma\left(X_1^o, I\right)$
11: end if
12: Broadcast $\Xi$ to all nodes
13: Set $\theta_r \leftarrow 0$, $\pi^o \leftarrow -1$, $c_\delta \leftarrow 0$
14: for $j \in \left\{\kappa + r + (j-1)N_n : j \in \mathbb{Z}_+, 1 \leq j \leq 1 + \left\lfloor \frac{N_{VM} - r - \kappa}{N_n} \right\rfloor \right\}$ and $c_\delta < M_\delta$ do
15:    Set $X \leftarrow \vec{0}$, $\xi \leftarrow \Xi_{1:j}$, $X_\xi \leftarrow 1$
16:    Compute $Y \leftarrow F(X)$
17:    if $\theta_r < \theta(Y)$ then
18:        Set $\theta_r \leftarrow \theta(Y)$, $\pi^o \leftarrow$ j
19:    else
20:        if $\theta_r < \delta\theta(Y)$ then
21:            Compute $c_\delta \leftarrow c_\delta + 1$
22:        else
23:            Break
24:        end if
25:    end if
26: end for
27: Reduce $\theta^o \leftarrow \max_r \theta_r$ to node $1$ and receive corresponding $\pi^o$
28: if r=1 then
29:    Set $X^{opt} \leftarrow \vec{0}$, $\xi \leftarrow \Xi_{1:\pi^o}$, $X_\xi^{opt} \leftarrow 1$
30: end if
```

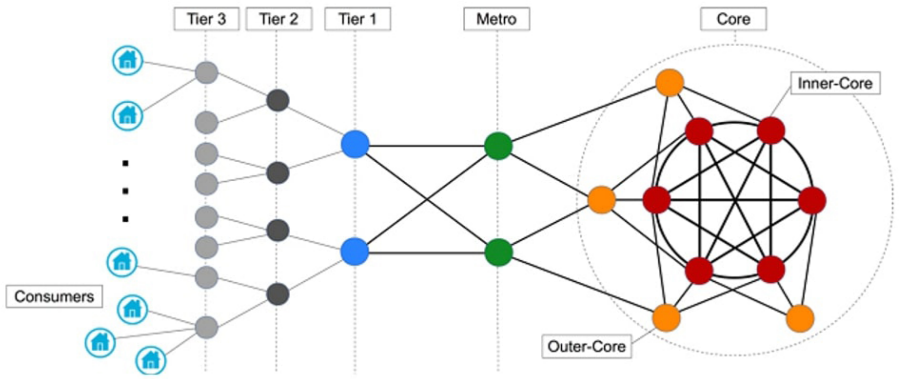


Figure 3: BT network infrastructure composed of core sites, metro sites, and MSAN sites.

Table 2: Characteristics of the vCDN.

Level	**Sites**	**Average Connections per Site**	**Total Connections**	**Average Bandwidth per Site (Gbps)**	**Average Bandwidth per Connection (Gbps)**
1	8	26.6250	213	1922.5000	72.2066
2	12	14.5000	174	791.6667	54.5977
3	86	4.7558	409	228.6047	48.0685
4	1026	1.4815	1520	42.6355	28.7789

the last level forward requests to level 3 through constituent sites at the lowest level. Another important observation is that only levels 1 and 4 have intra-level connections, with level 1 having all-to-all intra-level connections. This is performed to enable forwarding to any site in order to minimize rejection of tasks due to a lack of computational resources.

Each site of the vCDN hosts two nodes with 8 cores, 4 TB of RAM memory and 10 TB of storage. The VMs, as well as the requests, were of a single content type. Each VM has 2 Cores, 16 GB of RAM and 500 GB of storage and the chance of a cache hit was 60% [8]. The weights vector was chosen to be $\vec{w} = [3.0\ 1.0\ 2.0\ 2.0]$ for vCPU, Memory, Storage and Network requirements [8].

Table 3: Intervals for request requirements.

Requirements	**Intervals**
D (seconds)	$[300, 1000]$
C (vCPU)	$[0.01, 0.02]$
M (GB)	$[0.04, 0.05]$
S (GB)	$[0.8, 1.0]$
N (Gbps)	$[0.005, 0.006]$

The requirements of the requests were computed in the intervals given in Tab. 3. The requests were generated at the bottom-most level (level 4) following a uniform random distribution within prescribed minimum and maximum number of requests. Furthermore, the vCDN was simulated for 3600 seconds with a time-step of 1 second. The choice of parameters has been performed arbitrarily to represent requests spanning between 300 to 1000 seconds and does not affect applicability of the proposed scheme.

The simulations were executed on the ARIS supercomputer (GRNET). Each compute node consists of 2× Ivy Bridge - Intel Xeon E5-2680v2 (10 cores each) and 64GB RAM. Two tasks were executed per node with 10 cores (threads) per task.

5.2. Performance results

In order to assess the performance of the proposed two-phase optimization scheme, various scenarios were examined. Initially, the performance was assessed by executing the proposed scheme for different amount of requests. The values of the parameters related to stagnation of the second phase of the proposed scheme were set to $\delta = 0.95$ and $M_\delta = 10$. Each node executes $N_p = 20$ simulations. The choice of these parameters has been performed after extensive experimentation. The two-phase scheme was executed in 32 nodes. The maximum value for allowed VM placements was not set, thus it was considered equal to the the number of sites $N_{VM} = N_s = 1132$.

The performance and the number of VMs corresponding to the final placement for various numbers of approximate incoming requests is given in Fig. 4a and Fig. 4b. The execution time of the proposed scheme increases analogously to the number of input requests as can be observed by Fig. 4a. However, the increase is not linear, since, beyond 176200 input requests, the vCDN starts to reject incoming requests due to depletion of available resources (with respect to VM capacity). Thus, elapsed time increases at a slower pace.

Similarly, the number of VMs in the final placement, for each experiment, also increases. The large increase is a consequence of the chosen relaxed termination criterion thus allowing a large number of steps before stagnation terminates the iterative process. A more relaxed termination criteria may lead to an increased number of VMs by over-provisioning to avoid rejection of input requests. However, this increased number of VMs leads to placements that favour service delivery. The use of more complex $\theta(Y)$ score functions on the second phase substantially affects the final placement. By fixing parameters on the proposed scheme more deterministic results are expected.

5.3. Scalability results

The scalability of the proposed scheme was assessed by examining its performance and speedup for a range of computational nodes and available processing cores while keeping the number of simulations constant. The number of input requests was set to ≈ 3283000 requests. The performance and speedup of the proposed two-phase scheme are given in Fig. 5 and Fig. 6. The speedup of the proposed scheme is close to the ideal (linear) for the available number of nodes and processing cores with respect to a fixed number of simulations (equal to 1280), since the communication overhead is limited compared to the required computations. The simulations are distributed equally to the available tasks and consequently to the nodes.

5.4. Effect of the path formation strategy

We also conducted a set of experiments to assess the effect of the path formation strategy in the placements. The first path formation strategy is described in Alg. 1 and the second is based only on max available bandwidth (maxBW). In order to avoid interference in the results, the inputs described in Tab. 3 where fixed to the average of the intervals. Moreover, the sequence of input requests (≈ 36500) was fixed and distributed uniformly to the sites at the lowest level (level 4). The experiments were executed in 32 nodes. Each node executed 2 tasks with 10 available cores per task. The number of simulations per task for the first phase was set to 20, as in previous experiments.

The histogram of scores $\eta(Y)$ for the two path formation strategies is presented in Fig. 7. Moreover, the average score per level of the vCDN with respect to the two path formation strategies is given in Tab. 4. In the two histograms it is evident that apart from the sites at the top most level whose score was explicitly set to 1; the scores of the other sites were in the interval $[0.0727, 0.4717]$ for the path formation strategy described by Alg. 1 and in the interval $[0.0732, 0.4809]$ for the maxBW path formation strategy. For both cases, the majority of sites has a score $\eta(Y)$ in the interval $[0.25, 0.30]$ with the maxBW having more sites with scores above 0.25. This can be also seen from the average scores per level in Tab. 4. The increase in terms of average score $\eta(Y)$ is caused by the load balancing approach followed by the maxBW path formation strategy, which guides tasks to sites with most available network resources at a given moment. This approach guides more tasks to links with more total bandwidth that are more likely to have increased utilization when hosting VMs. Furthermore, higher total bandwidth connections usually lead to sites with more connections at their immediate upper level. In the case of Alg 1, a maxBW approach is followed only if there is no VM with available resources, otherwise the site with available resources receives the request. This may lead to uneven load among constituent sites retaining VMs and thus an overall reduced score, since computation of the score is related to global metrics of the vCDN. The maxBW approach is more effective for uniform workloads.

Table 4: Average score $\eta(Y)$ per level of the vCDN.

Level	$avg(\eta(Y))$ **(Alg. 1)**	$avg(\eta(Y))$ **(maxBW)**
1	1.0000	1.0000
2	0.2835	0.2874
3	0.2727	0.2763
4	0.2787	0.2824

The two-phase optimization scheme for the two path formation strategies leads to the same number of placements (840), with respect to score function $\theta(Y)$. However, for the maxBW path formation strategy, near optimal placement is achieved earlier as can be seen from Fig. 8. The near optimal number of placements with score (≥ 0.99) is achieved in the the case of Alg. 1, with 748 VM placements. While for the maxBW path formation strategy, the near optimal number of placements is achieved with 728 VM placements. For both path formation strategies convergence to the optimal value for the score function $\theta(Y)$ is monotonic. It should be noted that for the experiment with Alg. 1 as path formation strategy, the performance was 97.2338 seconds, while the performance of the maxBW path formation strategy was 98.2581 seconds. The convergence behavior of the proposed two-phase scheme, with the maxBW path formation strategy, is assessed using the norm $\|(X_1^o)^k - (X_1^o)^{k-1}\|_2$ where $k-1$ and k denote two consequent executions of the two-phase scheme with the later executing twice the number of simulations with random placements on the first phase. The first ranking $k = 1$ was computed with 40 executed simulations with random placements during first phase, while the last ranking $k = 8$ was computed with 5120 executed simulations with random placements during the first phase. The convergence behavior of the site ranking with respect to the number of executed simulations during the first phase with maxBW path formation strategy is given in Fig. 9. The case with $k = 1$ is not depicted in the figure and is used to initiate the computation of the norm. From Fig. 9 it can be seen that the number of simulations during first phase affects the ranking. However, progress towards convergence for computing a ranking is faster when the total number of simulations is relatively small and gradually slows down as the total number of simulations increases. Thus, a very large total number of simulations with random placements during the first phase might be proven ineffective in the case of very large number of simulations. On the other hand a very small number might lead to a placement that is very far from optimal. The number of placements even in the case of the largest number of simulations is 840.

A Monte Carlo approach can be used to derive an optimal placement. In this approach each processing node executes several simulations with random placements retaining the one requiring less VMs while minimizing rejected requests. The optimal placements are collected to the head node where the derived local optimal placements are rerun to obtain the best one. For 32 nodes and 20 simulations per node with the same setup $(640 + 32$ total simulation) and maxBW path formation strategy, the Monte Carlo approach lead to 859 placements requiring 50.656 seconds. For 40 simulations per node $(1280 + 32$ total simulation) the optimal placement included 846 placements requiring 102.375 seconds. For 80 simulations per node $(2560 + 32$ total simulation) the optimal placement included 841 placements requiring 208.012 seconds. Thus, in order to obtain a similar result to the one computed with the proposed scheme more simulations are required, increasing computational work. Moreover, the optimal placement with the Monte Carlo approach depends on the total number of simulations executed, due to the random generation of the placements. A small number may lead to increased number of placements, since the number of VMs is chosen randomly. However, the Monte Carlo approach requires less overall communications.

6. Conclusion

In this paper, we presented an update and improvement to a previously presented DTS framework for vCDNs, which enables more general and accurate parallel simulations to be carried out while retaining the performance advantages of the previous iteration. This framework was combined with a two-phase optimization scheme for estimating optimal placements in vCDN networks. This scheme enables the simulation-based optimization of large scale vCDN networks with respect to functional and non-functional requirements described as mathematical functions and included in a score function. Initially the ranking of sites is performed, followed by selection of the appropriate number of placements to service a predefined distribution of requests. The proposed scheme was assessed on a supercomputing environment, using a real world vCDN architecture data obtained by BT, and was proven scalable up to a large number of requests and simulations. The proposed two-phase scheme was used to analyze the effect of different path formation strategies for the optimal placement of caches in order to assess the effect of different paths to final placement, where a maximum bandwidth approach led to faster convergence. Furthermore, the proposed optimization scheme can be used in conjunction with other simulation frameworks, for example based on Discrete Event Simulation, taking into account more phenomena in the micro level.

Future work includes the acceleration of the convergence to an optimal placement as well as performance improvements related to the amount of simulations required to produce site ranking.

Acknowledgements

The authors acknowledge the Greek Research and Technology Network (GRNET) for the provision of the National HPC facility ARIS under project PR006053-ScaleSciCompIII. This work is funded by the European Unions Horizon 2020 Research and Innovation Programme through RECAP (`https://recap-project.eu/`) under Grant Agreement Number 732667.

The authors acknowledge Peter J. Willis for the provision of BT data.

References

[1] Cisco, Cisco Visual Networking Index: Forecast and Methodology, 2016-2021, URL `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html`, [Online; accessed 09-Aug-2018], 2017.

[2] A. Vakali, G. Pallis, Content delivery networks: Status and trends, IEEE Internet Computing 7 (6) (2003) 68–74.

[3] G. Pallis, A. Vakali, Insight and perspectives for content delivery networks, Communications of the ACM 49 (1) (2006) 101–106.

[4] A. Moreira, J. Moreira, D. Sadok, A. Callado, M. Rodrigues, M. Neves, V. Souza, P. P. Karlsson, A case for virtualization of Content Delivery Networks, in: Proceedings of the 2011 Winter Simulation Conference (WSC), ISSN 0891-7736, 3178–3189, doi:10.1109/WSC.2011.6148016, 2011.

[5] T.-W. Um, H. Lee, W. Ryu, J. K. Choi, Dynamic Resource Allocation and Scheduling for Cloud-Based Virtual Content Delivery Networks, ETRI Journal 36 (2) (2014) 197–205.

[6] C.-F. Lin, M.-C. Leu, C.-W. Chang, S.-M. Yuan, The study and methods for cloud based CDN, in: Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on, IEEE, 469–475, 2011.

[7] M. Wang, P. P. Jayaraman, R. Ranjan, K. Mitra, M. Zhang, E. Li, S. Khan, M. Pathan, D. Georgeakopoulos, An overview of cloud based content delivery networks: research dimensions and state-of-the-art, in: Transactions on Large-Scale Data-and Knowledge-Centered Systems XX, Springer, 131–158, 2015.

[8] C. K. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, P. T. Endo, D. Tzovaras, S. Svorobej, T. Lynn, Simulating large vCDN networks: A parallel approach, Simulation Modelling Practice and Theory 92 (2019) 100 – 114, ISSN 1569-190X, doi:https://doi.org/10.1016/j.simpat.2019.01.001, URL `http://www.sciencedirect.com/science/article/pii/S1569190X19300012`.

[9] N. Herbaut, D. Negru, D. Dietrich, P. Papadimitriou, Dynamic deployment and optimization of virtual content delivery networks, IEEE MultiMedia 24 (3) (2017) 28–37.

[10] J. Domaschka, Deliverable 3.1. Initial Requirements, Tech. Rep., RECAP Project, URL `https://recap-project.eu/about/public-deliverables/`, 2017.

[11] G. Figueira, B. Almada-Lobo, Hybrid simulation–optimization methods: A taxonomy and discussion, Simulation Modelling Practice and Theory 46 (2014) 118–134.

[12] J. Xu, E. Huang, C.-H. Chen, L. H. Lee, Simulation optimization: A review and exploration in the new era of cloud computing and big data, Asia-Pacific Journal of Operational Research 32 (03) (2015) 1550019.

[13] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, O. C. M. B. Duarte, Orchestrating virtualized network functions, IEEE Transactions on Network and Service Management 13 (4) (2016) 725–739.

[14] W. Rankothge, F. Le, A. Russo, J. Lobo, Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms, IEEE Transactions on Network and Service Management 14 (2) (2017) 343–356.

[15] T. Ecarot, D. Zeghlache, C. Brandily, Consumer-and-Provider-Oriented Efficient IaaS Resource Allocation, in: Parallel and Distributed Processing

Symposium Workshops (IPDPSW), 2017 IEEE International, IEEE, 77–85, 2017.

[16] J. Xu, J. Fortes, A multi-objective approach to virtual machine management in datacenters, in: Proceedings of the 8th ACM international conference on Autonomic computing, ACM, 225–234, 2011.

[17] J. Xu, Autonomic application and resource management in virtualized distributed computing systems, University of Florida, 2011.

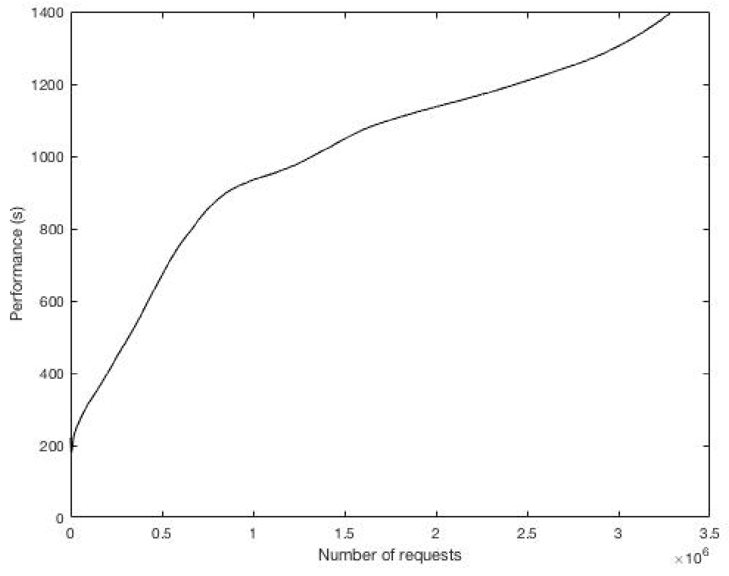
[18] L. Yala, P. A. Frangoudis, G. Lucarelli, A. Ksentini, Cost and availability aware resource allocation and virtual function placement for CDNaaS provision, IEEE Transactions on Network and Service Management 15 (4) (2018) 1334–1348.

[19] H. Ibn-Khedher, M. Hadji, E. Abd-Elrahman, H. Afifi, A. E. Kamal, Scalable and cost efficient algorithms for virtual CDN migration, in: Local Computer Networks (LCN), 2016 IEEE 41st Conference on, IEEE, 112–120, 2016.

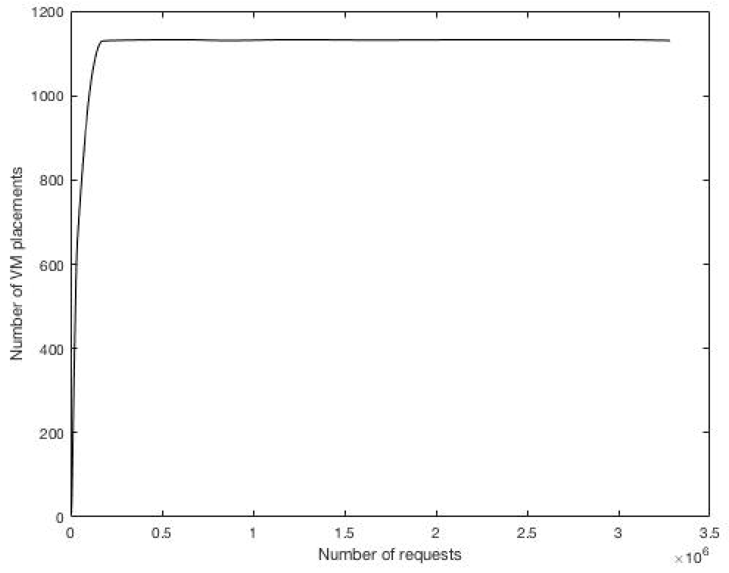
[20] H. Ibn-Khedher, E. Abd-Elrahman, A. E. Kamal, H. Afifi, OPAC: An optimal placement algorithm for virtual CDN, Computer Networks 120 (2017) 12–27.

[21] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, second edn., doi:10.1137/1.9780898718003, 2003.

[22] ISO/IEC, ISO International Standard ISO/IEC 14882:2011(E) Programming Language C++ .



(a)



(b)

Figure 4: Performance (a) and number of VM placements (b) for various numbers of input requests for the proposed two-phase optimization scheme.

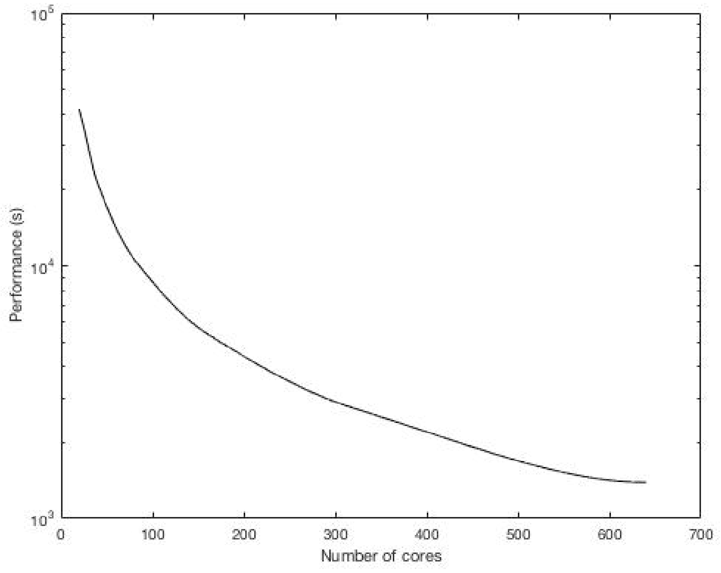


Figure 5: Performance of the proposed two-phase optimization scheme for various numbers of cores.

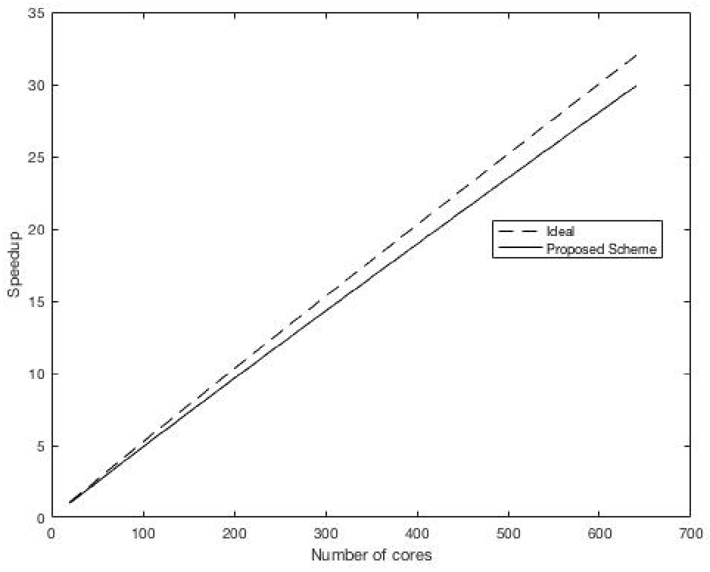
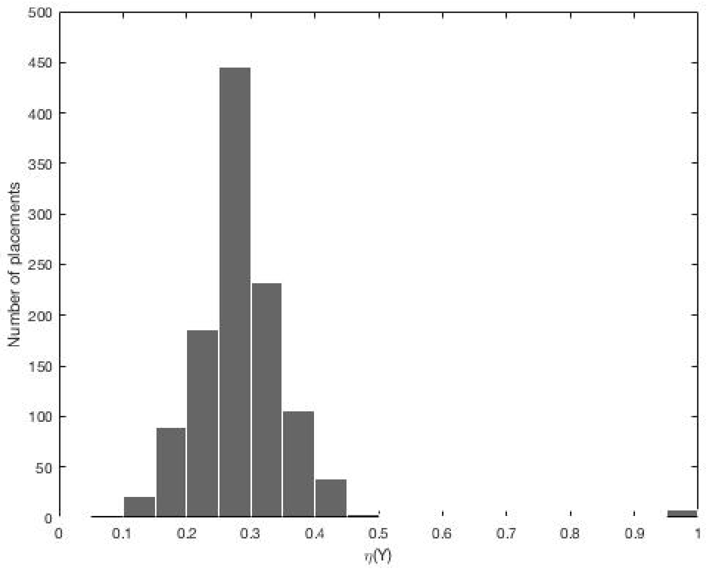
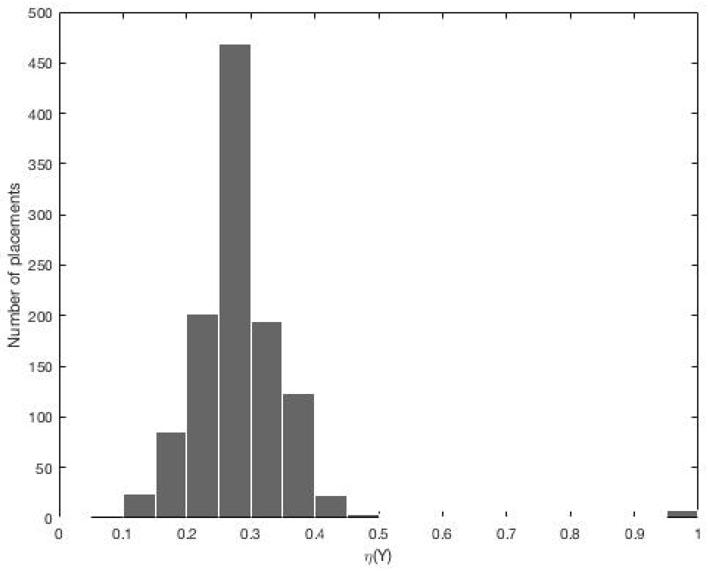


Figure 6: Speedup of the proposed two-phase optimization scheme for various numbers of cores.

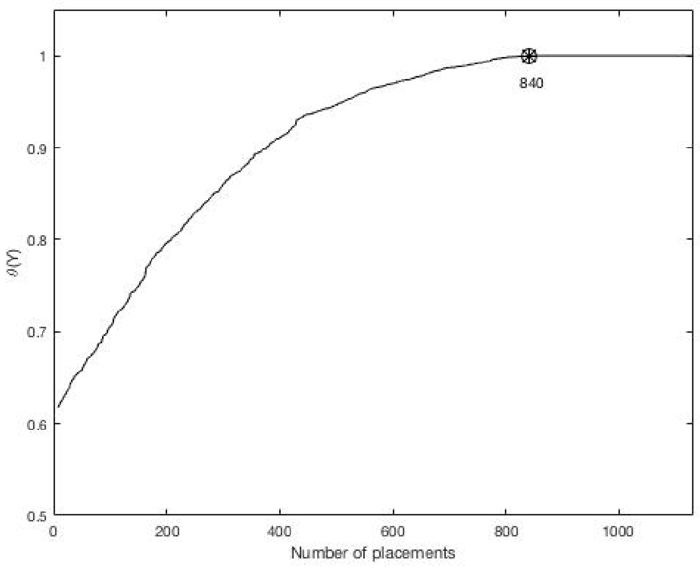


(a)

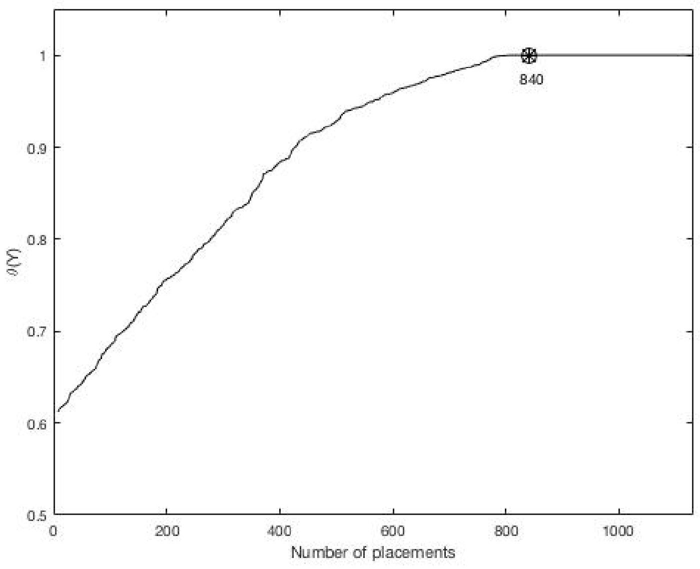


(b)

Figure 7: Histogram of number of placements with respect to score $\eta(Y)$ for path formation based on (a) Alg. 1 and (b) maxBW.



(a)



(b)

Figure 8: Score $\theta(Y)$ with respect to number of placements for path formation based on (a) Alg. 1 and (b) maxBW.

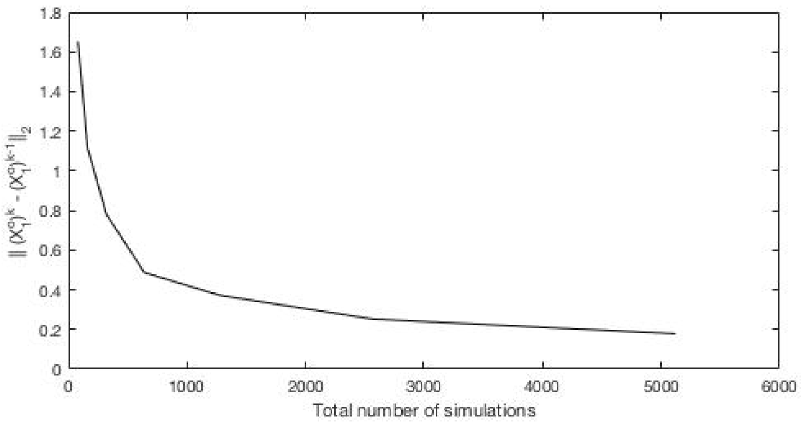


Figure 9: Convergence behavior of the ranking of sites with respect to the number of executed simulated during first phase with maxBW path formation strategy.